Proposta de heurística para o K-Means com base no cálculo da Distância de Manhattan

Arthur Rodrigues Soares Quadros[1], **João Pedro Torres**[1], **Sarah Luiza de Souza Magalhães**[1]

[1] Pontifícia Universidade Católica de Minas Gerais

{aquadros,joao.silva,sarah.magalhaes.1280966}@sga.pucminas.br

Resumo. *Proposta de heurística customizada para solução de problemas reais de clusterização utilizando o algoritmo de Inteligência Artificial para aprendizado não supervisionado* K-Means*. Para isso, foi criado um algoritmo de clusterização capaz de agrupar dados bidimensionais, que possui a mesma proposta do* K-Means*, porém, com algumas mudanças em relação ao modo de seleção de centróides iniciais, cálculo de distância entre pontos e definição de convergência com uso de um percentual de tolerância. Os resultados finais, dentro do estudo de caso proposto, se mostram satisfatórios dada a simplicidade da heurística em comparação à implementada na biblioteca* Python Scikit-learn*. A Soma do Quadrado dos Erros (SSE) dessa heurística fica maior no total, porém, o erro por* cluster*, em alguns casos, fica muito menor.*

Palavras-chave: K-Means*; Clusterização; Heurísticas; Distância de Manhattan.*

1. Introdução

Dentre os problemas ambientais mais expressivos no cenário nacional, destaca-se a exorbitante quantidade de lixo eletrônico descartada incorretamente. Diante desse cenário, a implementação de unidades de tratamento mostra-se como uma solução viável para o problema. Contudo, restringindo o contexto de ambientação da proposta ao estado de Minas Gerais, nota-se a necessidade de mapeamento das regiões e municípios mais carentes dos serviços de tratamento especial de lixo eletrônico, haja vista que a quantidade e o posicionamento dessas estações não é conhecido. A partir dessa conjuntura, a proposta deste artigo, consistente no uso de algoritmos de Inteligência Artificial e estratégias de Aprendizado de Máquina não-supervisionado, visa realizar o agrupamento dos municípios mineiros em regiões de possível instalação de estações de tratamento especial de lixo.

2. Metodologia

A implementação da heurística, metodologia para solução de um problema [Romanycia and Pelletier 1985], foi feita na linguagem de programação *Python*, tendo sido escolhida por sua simplicidade de uso e abundância de recursos já nativos ou de rápida implementação. Todo o código, testes e visualização de resultados foram feitos através de suas bibliotecas externas, notavelmente: *Scikit-learn, Numpy, Pandas* e *Matplotlib*.

A proposta da heurística surge com a ideia de simplificar a execução do algoritmo *K-Means* com resultados aceitáveis para o contexto do problema em que, principalmente,

não haja uma perda de representatividade em relação ao revés, nem à solução do algoritmo padrão usado para comparações. O algoritmo base utilizado para a avaliação de desempenho da heurística é o proposto na biblioteca *Scikit-learn*.

2.1. Heurística propriamente dita

A heurística consiste em alguns procedimentos em diferentes etapas do algoritmo padrão do *K-Means*, em grande parte baseadas no cálculo da Distância de Manhattan, por ser simples de aumentar o número de dimenções, ideal para clusterização [Pandit et al. 2011], por isso, ele não deixa de ser uma variação desse algoritmo através de algumas abordagens matemáticas simplificadas. Essas etapas são: definição dos centróides iniciais, cálculo da distância propriamente dita para seleção dos pontos para cada cluster e a condição de parada utilizando uma métrica por tolerância.

Na seleção dos centróides iniciais, essa heurística seleciona N grupos de centróides diferentes com posições aleatórias, isto é, seleciona K (número de clusters) pontos P pertencentes ao conjunto de pontos X contido nos dados, para depois montar N matrizes de afinidade simplificadas (utilizando o cálculo de Distância de Manhattan ao invés do Kernel de Gauss) entre todos eles, uma para cada grupo de centróides selecionados aleatoriamente. Após esse cálculo, temos a métrica de escolha de qual dos N grupos serão utilizados para início do cálculo, para isso, em cada uma das matrizes $K \times K$ geradas, calculamos a distância média entre todos os pontos em cada uma das matrizes, selecionando o grupo que possui a maior distância média de todas para serem os centróides iniciais. Essa ideia segue a premissa de que, quanto maior a distância entre centróides, melhor a área coberta por cada centróide tende a ser distribuída.

Após isso, com o grupo de centróides ideal escolhido, nós realizamos a primeira iteração para gerar os K *clusters* e, com eles definidos, nós recalculamos os centróides com uma média aritmética simples entre todas as coordenadas (x, y) pertencentes a cada *cluster*. Na primeira iteração, o teste para condição de parada não é realizado, mas, após atualizar os centróides uma primeira vez e iterar uma segunda vez, os valores pertencentes a cada *cluster* são atualizados e comparados com os resultados anteriores. Nesse momento, o teste de convergência é realizado. A tolerância T é um percentual que determina o quanto os valores precisam mudar para ser considerado uma mudança significativa, sendo que uma tolerância muito alta permite que dados com valores muito distantes ainda sejam considerados IGUAIS, enquanto uma tolerância muito baixa considerará que dados muito próximos ainda sejam considerados DIFERENTES. Seja v_o a distância entre um dado *cluster* e um ponto x qualquer da iteração antiga e v_n, de forma análoga, da iteração nova, para v_n ser considerado igual, ele precisa estar dentro de um intervalo fechado $[v_o(1-T), v_o(1+T)]$. Caso mais de 75% dos dados atualizados sejam considerados iguais aos da iteração anterior de acordo com a tolerância determinada, uma parada antecipada é chamada, o que faz o algoritmo parar a execução mesmo se ainda não houver alcançado o número de iterações máximo determinado. Em resumo, esse algoritmo com a tolerância baixa fará mais iterações, com a tendência de otimizar mais o resultado final, enquanto, com uma tolerância mais alta, ele tenderá a acabar com a parada antecipada mais cedo. Em outras palavras, a tolerância mais alta determina convergência mais rápido; a mais baixa, mais lento.

3. Resultados

Para a execução dos modelos, foram contemplados cinco hiper-parâmetros, atuantes, cada um, nas áreas de: aleatoriedade para seleção dos centróides iniciais (A); quantidade de grupos geográficos (*clusters*) a serem gerados (B); máximo de iterações com redefinição dos *clusters* e recálculo dos centróides (C); na verificação de desempenho para avaliação da parada antecipada (D) e na quantidade máxima de iterações (E). Partindo da utilização desses recursos, ambos os algoritmos foram executados de forma equiparável, de modo que os resultados obtidos por ambos pudessem ser comparados de forma justa. Tais hiper-parâmetros podem ser visualizados na seguinte tabela.

nome	**valor**	**atuação**
n_clusters	10	B
n_init	50	A
max_iter	5000	C, E
tol	$1 \cdot 10^{-5}$	D, E
random_state	5	A

Table 1. Hiper-parâmetros e suas funções

Após executados os modelos, duas notórias diferenças foram observadas. A primeira, de cunho gráfico, que pode ser facilmente notada a partir dos gráficos seguintes, diz respeito à diferença de centralidade dos centróides resultantes em relação aos seus respectivos *clusters*. A segunda, de cunho analítico, trata da diferença entre os valores J, dados pela expressão $\sum_{c=1}^{k} \sum_{x_j \in C_c} d(x_j,\ \bar{x}_c)^2$, de cada um dos modelos, que denotam a SSE calculada de cada um deles.

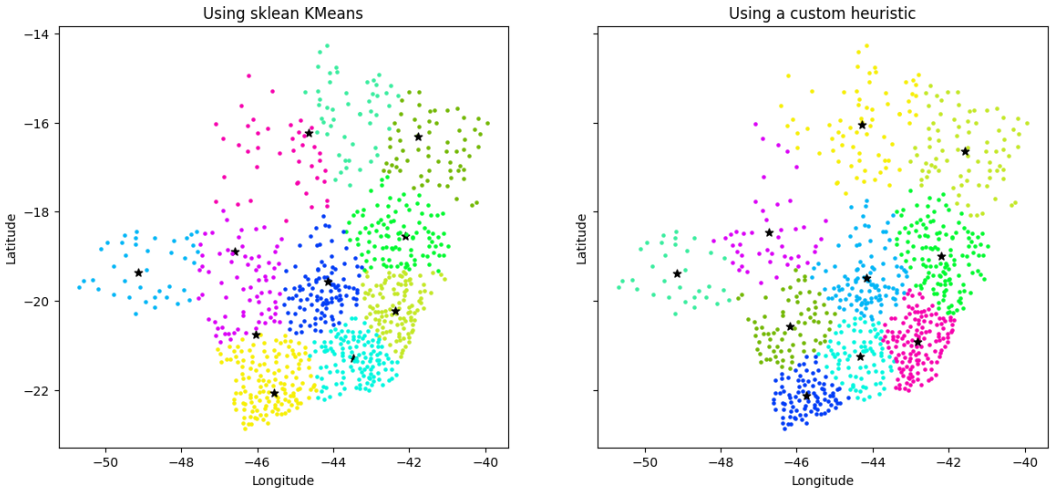

Figure 1. Comparação dos resultados

Ao avaliar essa diferença, nota-se que o modelo com essa heurística customizada gerou um J equivalente a aproximadamente 103,35% do SSE calculado pelo modelo base, o que mostra que o primeiro conseguiu, com uma heurística consideravelmente mais simples, um resultado matematicamente muito próximo da base, considerando-se a métrica de erro total. Além disso, há também o SSE calculado para cada *cluster* individualmente,

em que o erro individual médio no algoritmo base tende a ser melhor distribuído, com distâncias que variam entre 43,2 e 96,6, e valores intermediários muito abaixo da média entre os dois extremos, enquanto, nessa heurística, as distâncias variam entre 28,8 e 102,4, e valores intermediários levemente abaixo da média, o que significa que, apesar de que alguns *clusters* possuem um valor consideravelmente menor do que o mínimo do algoritmo base, há outras distâncias que aumentam o erro total a ponto de ficar maior do que a base. Essa heurística consegue diminuir o erro individual de alguns *clusters* muito bem, mas à custa do erro individual de outros serem aumentados.

References

Pandit, S., Gupta, S., et al. (2011). A comparative study on distance measuring approaches for clustering. *International journal of research in computer science*, 2(1):29–31.

Romanycia, M. H. and Pelletier, F. J. (1985). What is a heuristic? *Computational intelligence*, 1(1):47–58.